Optimization for Flexible Job Shop Scheduling by Evolutionary Representation

K.Bharathi\textsuperscript{1} and C.Vijaylakshmi\textsuperscript{2}

\textsuperscript{1} Department of Mathematics, SCSVMV University
Kanchipuram-631561, Tamil Nadu, India
Email: 03bharathi@gmail.com

\textsuperscript{2} SAS, Mathematics Division, VIT University
Chennai- 600 127, Tamil Nadu, India.
Email: vijusesha2002@yahoo.co.in

Received 20 June 2014; accepted 4 August 2014

Abstract. In this paper we have discussed a flexible job-shop scheduling problem by considering the cases as the assignment of each operation to a machine, and the other is the scheduling of this set of operations in order to minimize our criterion (e.g. the make span and completion time of each job). After applying the operators like crossover and mutation criterion where minimized. Here, we propose effective genetic encodings, such as job and machine representations as matrices of the chromosome, and Genetic operators where associated with these representations.

Keywords: Flexible job-shop scheduling, Optimization, Evolutionary Algorithms, Genetic representation, Encoding matrices, Makespan.

AMS Mathematics Subject Classification (2010): 05C78

1. Introduction

Job shop scheduling problems (JSS) are computationally complex problems. Because JSS are NP-hard i.e., they can’t be solved within polynomial time force or undirected search methods are not typically feasible, at least for problems of any size. Thus JSS tend to be solved using a combination of search and heuristics to get optimal or near optimal solutions. The term ‘Scheduling’ in manufacturing systems is used to the determination of the sequence in which parts are to be processed over the production stages, followed by the determination of the start-time and finish-time of processing of parts, so as to meet an objective or a set of objectives.

Scheduling plays a crucial role to increase the efficiency and productivity of the manufacturing system. The scheduling can be classified into (i) Single machine scheduling (ii) Flow shop scheduling (iii) Job shop scheduling. Optimisation methods attempt to find the optimal solution through mathematical programming techniques or methods [5-7]. However, mathematical programming methods are time-consuming, and thus, many researchers focus on developing heuristic algorithms [11-15], algorithms in common use include shifting bottleneck (SB) [15], Tabu search (TS) [10], simulated annealing (SA) [9], the genetic algorithm (GA) [8], artificial immune system (AIS) [16],
K.Bharathi and C.Vijayalakshmi

and modified Particle swarm optimization (PSO) [17]. In [19] proposed a local search genetic algorithm that uses an efficient solution representation strategy in which both checking of the constraints and repair mechanism can be avoided.

2. Structure of the scheduling problem

Consider a set of ‘n’ jobs \{J_i\}, 1 \leq i \leq n; these jobs are independent of one another. Each job ‘J_i’ has an operating sequence, called \(P_i\). Each operating sequence \(P_i\) is an ordered series of \(X_i\) operations, \(O_j\) indicating the position of the operation in the technological sequence of the job. The realization of each operation \(O_j\) requires a resource or a machine selected from a set of machines, \(\{M_k\}\), 1 \leq k \leq m; ‘m’ is the total number of machines existing in the shop, this implying the existence of an assignment problem. There is a pre-defined set of processing times; for a given machine, and a given operation, the processing time is denoted by \(T_{ijM}\). An operation which has started runs to completion (non-preemption condition). Each machine can perform operations one after another (resource constraints). The time required to complete the whole job constitutes the make span \(C_{max}\). The time required to complete each job is as \(C_{T_i}\) our objective is to determine the set of completion times of all jobs to minimize \(C_{max}\) and also to minimize \(C_{T_i}\).

3. Representation of the solution

The chromosome is represented by a set of jobs and each job is a matrix which contains its assignment operations. These operations are represented by three terms. The first column is the order number of the machine in its operating sequence. The second is the starting time of the operation if its assignment on this machine. The third is the completion time of the operation if its assignment on this machine. That is

\[
J_K = \begin{bmatrix}
  j & S_{ij} & C_{ij} \\
  j' & S'_{ij} & C'_{ij}
\end{bmatrix}, \text{ where } k, i = 1, 2, ..., n \text{ and } j = 1, 2, ..., m.
\]

4. Numerical calculation

Three jobs and five machines are considered. The operating sequences of these jobs are as follows in table 1.

<table>
<thead>
<tr>
<th>J</th>
<th>O</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
</tr>
</thead>
<tbody>
<tr>
<td>(J_1)</td>
<td>(O_{1,1})</td>
<td>1</td>
<td>8</td>
<td>3</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>(O_{2,1})</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>(O_{3,1})</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>(J_2)</td>
<td>(O_{1,2})</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>(O_{2,2})</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>(O_{3,2})</td>
<td>9</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>(J_3)</td>
<td>(O_{1,3})</td>
<td>1</td>
<td>8</td>
<td>9</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>(O_{2,3})</td>
<td>5</td>
<td>9</td>
<td>2</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 1: The operating sequences
Optimization for Flexible Job Shop Scheduling by Evolutionary Representation

According to the machine used, the processing time of operations is described. One of the solution to the problem is as the matrix representation in fig. 1. Where $C_{\text{max}} = 11$

$$
\begin{bmatrix}
3 & 0 & 3 \\
3 & 3 & 5 \\
4 & 5 & 9
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 1 \\
1 & 1 & 5 \\
3 & 5 & 6
\end{bmatrix}
\begin{bmatrix}
5 & 0 & 2 \\
2 & 2 & 11 \\
0 & 11 & 11
\end{bmatrix}
$$

Figure 1: Example

5. Evolutionary representation

5.1. Initial population

The initial population is usually chosen at random. But in a combinatorial problem such as job shop scheduling, some constraints such as precedence and resources constraints must be satisfied. In this case, the binary representation is not convenient and chromosome syntax must be found to fit the problem. For these reasons, we have designed a matrix representation of the chromosome, and in order to create and to permit our set of solutions to evolve in a very large domain, we shall use a combination of some methods. We use a combination of the following priority rules. SPT: a high priority for the operation that has the Shortest Processing Time. LPT: a high priority for the operation that has the Longest Processing Time. LM: a high priority for the operation that permits to balance the load of the machine. Accordingly two solutions of the example (table 1) are taken as parent fig. 2 with $C_{\text{max}} = 11$ and fig.3 with $C_{\text{max}} = 13$.

$$
\begin{bmatrix}
3 & 0 & 3 \\
3 & 3 & 5 \\
4 & 5 & 9
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 1 \\
1 & 1 & 5 \\
3 & 5 & 6
\end{bmatrix}
\begin{bmatrix}
5 & 0 & 2 \\
2 & 2 & 11 \\
0 & 11 & 11
\end{bmatrix}
$$

Figure 2: Parent 1

$$
\begin{bmatrix}
3 & 0 & 3 \\
3 & 3 & 5 \\
4 & 5 & 9
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 1 \\
1 & 1 & 5 \\
3 & 5 & 6
\end{bmatrix}
\begin{bmatrix}
5 & 0 & 2 \\
2 & 2 & 11 \\
0 & 11 & 11
\end{bmatrix}
$$

Figure 3: Parent 2

5.2. Crossover operator

Crossover involves combining elements from two parent chromosomes into one or more child chromosomes. The role of the crossover is to generate a better solution by exchanging information contained in the current good ones. Here we use the following steps to get the offsprings

STEP 1: For child 1 job 1 of parent 1 is placed and jobs 2 and 3 of parent 2 is placed.

STEP 2: For child 2 job 1 of parent 2 is placed and jobs 2 and 3 of parent 1 is placed.

The offspring’s after crossover where represented as shown in the fig. 3 and fig. 4 where child 1 having $C_{\text{max}} = 11$ and child 2 having $C_{\text{max}} = 13$. 

$$
\begin{bmatrix}
3 & 0 & 3 \\
3 & 3 & 5 \\
4 & 5 & 9
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 1 \\
1 & 1 & 5 \\
3 & 5 & 6
\end{bmatrix}
\begin{bmatrix}
5 & 0 & 2 \\
2 & 2 & 11 \\
0 & 11 & 11
\end{bmatrix}
$$

$$
\begin{bmatrix}
4 & 3 & 6 \\
5 & 3 & 6 \\
3 & 2 & 4
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 3 \\
1 & 1 & 5 \\
3 & 5 & 6
\end{bmatrix}
\begin{bmatrix}
4 & 6 & 13 \\
2 & 6 & 11 \\
0 & 4 & 4
\end{bmatrix}
$$

$$
\begin{bmatrix}
4 & 3 & 6 \\
5 & 3 & 6 \\
3 & 2 & 4
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 3 \\
1 & 1 & 5 \\
3 & 5 & 6
\end{bmatrix}
\begin{bmatrix}
4 & 6 & 13 \\
2 & 6 & 11 \\
0 & 4 & 4
\end{bmatrix}
$$
5.3. Mutation

After crossover, each child produced by the crossover undergoes mutation with a low probability. The mutation is done as follows

1. Choose a job with the largest completion time $C_{T_i}$.
2. In that job $J_i$, replace the machine with the largest operating time by the least operating time.

The offspring's after mutation where represented as shown in fig.10 and fig.11 where mutation child 1 having $C_{\text{max}}=8$ and mutation child 2 having $C_{\text{max}}=5$.

<table>
<thead>
<tr>
<th>$J_1$</th>
<th>$J_2$</th>
<th>$J_3$</th>
<th>$J_1$</th>
<th>$J_2$</th>
<th>$J_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 0 3</td>
<td>4 0 3</td>
<td>5 0 6</td>
<td>1 0 1</td>
<td>1 1 2</td>
<td>5 0 2</td>
</tr>
<tr>
<td>4 3 5</td>
<td>5 3 6</td>
<td>3 6 7</td>
<td>4 1 7</td>
<td>1 2 4</td>
<td>2 4 13</td>
</tr>
<tr>
<td>4 5 8</td>
<td>2 6 11</td>
<td>0 7 7</td>
<td>4 7 13</td>
<td>3 4 5</td>
<td>0 13 13</td>
</tr>
</tbody>
</table>

Figure 3: Child 1  Figure 4: Child 2

After the variation operators like crossover and mutation we have got the solution with the makespan as 8 and 5 where the completion time for each job $C_{T_i}$ is also minimized.

6. Conclusion

Scheduling can be defined as a problem of finding an optimal sequence to execute a finite set of operations satisfying most of the constraints. The problem so formulated is extremely difficult to solve, as it comprises several concurrent goals and several resources which must be allocated to lead our goals, which are to maximize the utilization of individuals and/or machines and to minimize the time required to complete the entire process being scheduled. Here, we propose effective genetic encodings, such as job and machine representations as matrices of the chromosome, and Genetic operators where associated with these representations.

REFERENCES

Optimization for Flexible Job Shop Scheduling by Evolutionary Representation


10. S. Petrovic and C. Fayad, A genetic algorithm for job shop scheduling with load balancing, School of Computer Science and Information Technology, University of Nottingham, 2005.


K. Bharathi and C. Vijaylakshmi